Technical Cybersecurity

readelf

Reading ELF formatted files

Readelf v. objdump

- Readelf: displays information on ELF formatted files
- Objdump: displays information on object files

Similar but different

- Dump headers, symbols, relocations with both
- objdump: disassembly, source, debugging info
- readelf: section details, segments, sections

Why similar?

- ELF files are object files, and object files (on linux) are ELF files

When to use which?

READELF

- Sections and segments
- Symbols and function names
- Entry points

OBJDUMP

- Dissassembly
- …anything code related

Many tools show same information in different ways

What will it show?

ELF FILE HEADERS

- Magic: ELF files have the bytes 7f 45 4c 46 (7f ‘ELF’) at the start of the file
- Type, processor type, entry point

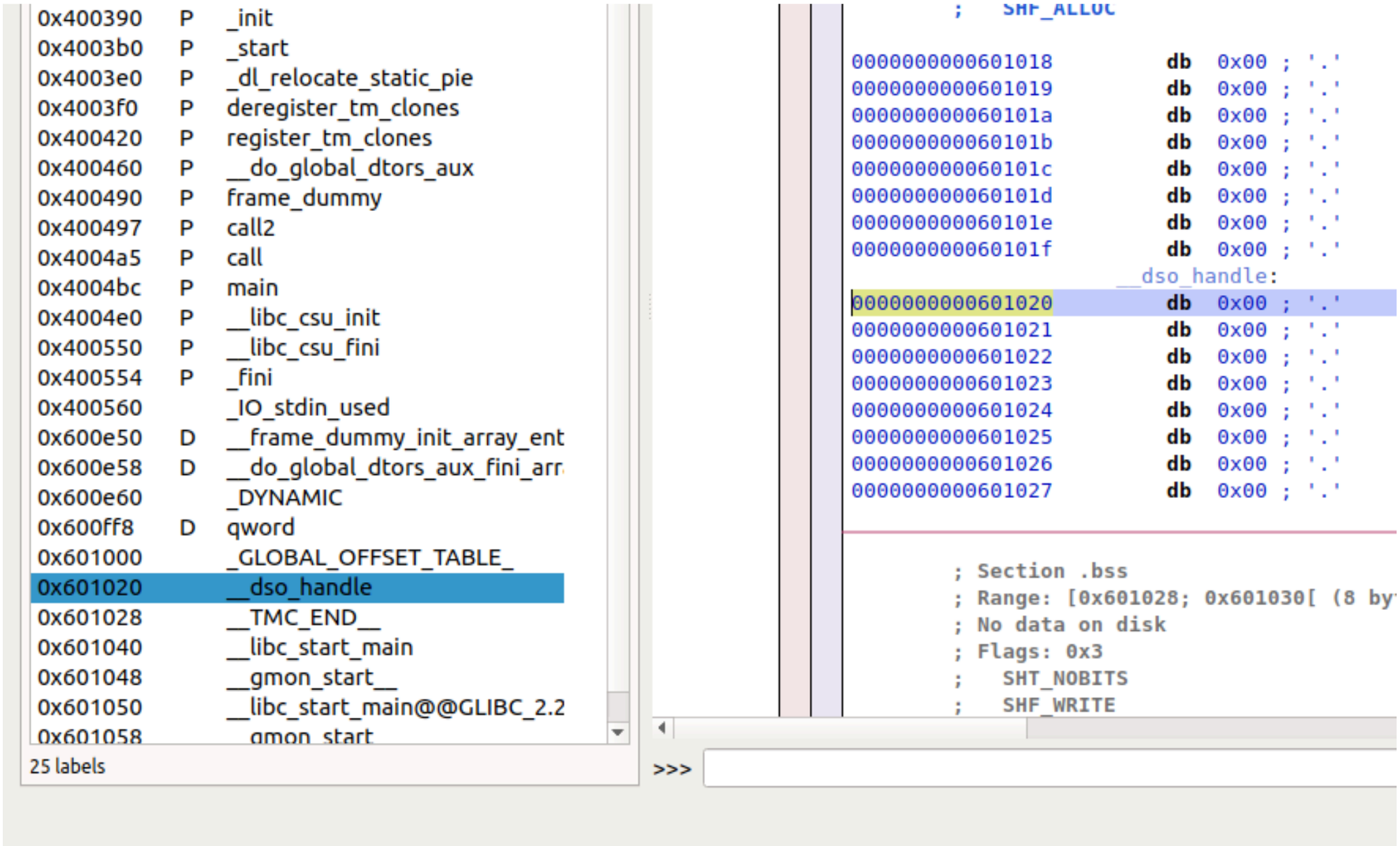
```
cclamb@ubuntu:~/Work/abi-playground $ readelf -h f2
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, littl
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file
  Machine:                           Advanced Micro Device
  Version:                           0x1
  Entry point address:               0x4003b0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          7160 (bytes into file
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         32
  Section header string table index: 31
cclamb@ubuntu:~/Work/abi-playground $
```

Symbols

Functions

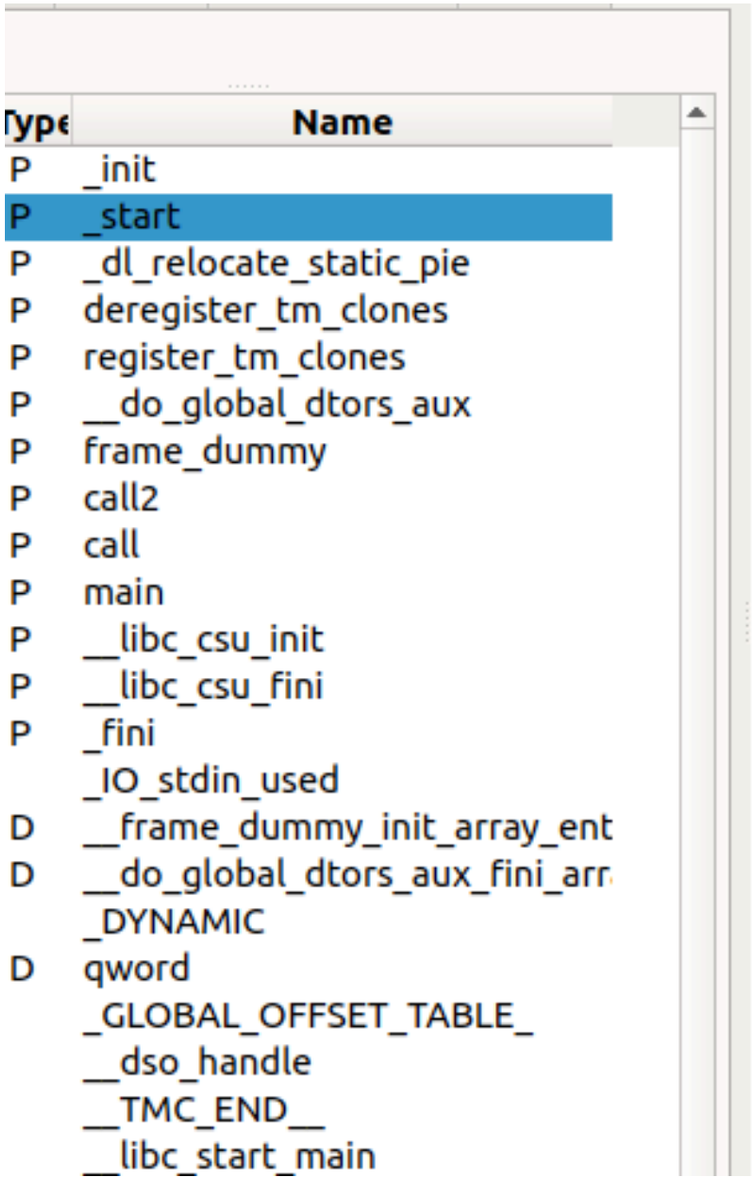
- **$ readelf -s f2**
  - You can see our functions
  - …and a bunch the compiler inserted for us
    - _start, _init, etc.
  - FUNC -> function
  - OBJECT, NOTYPE, FILE

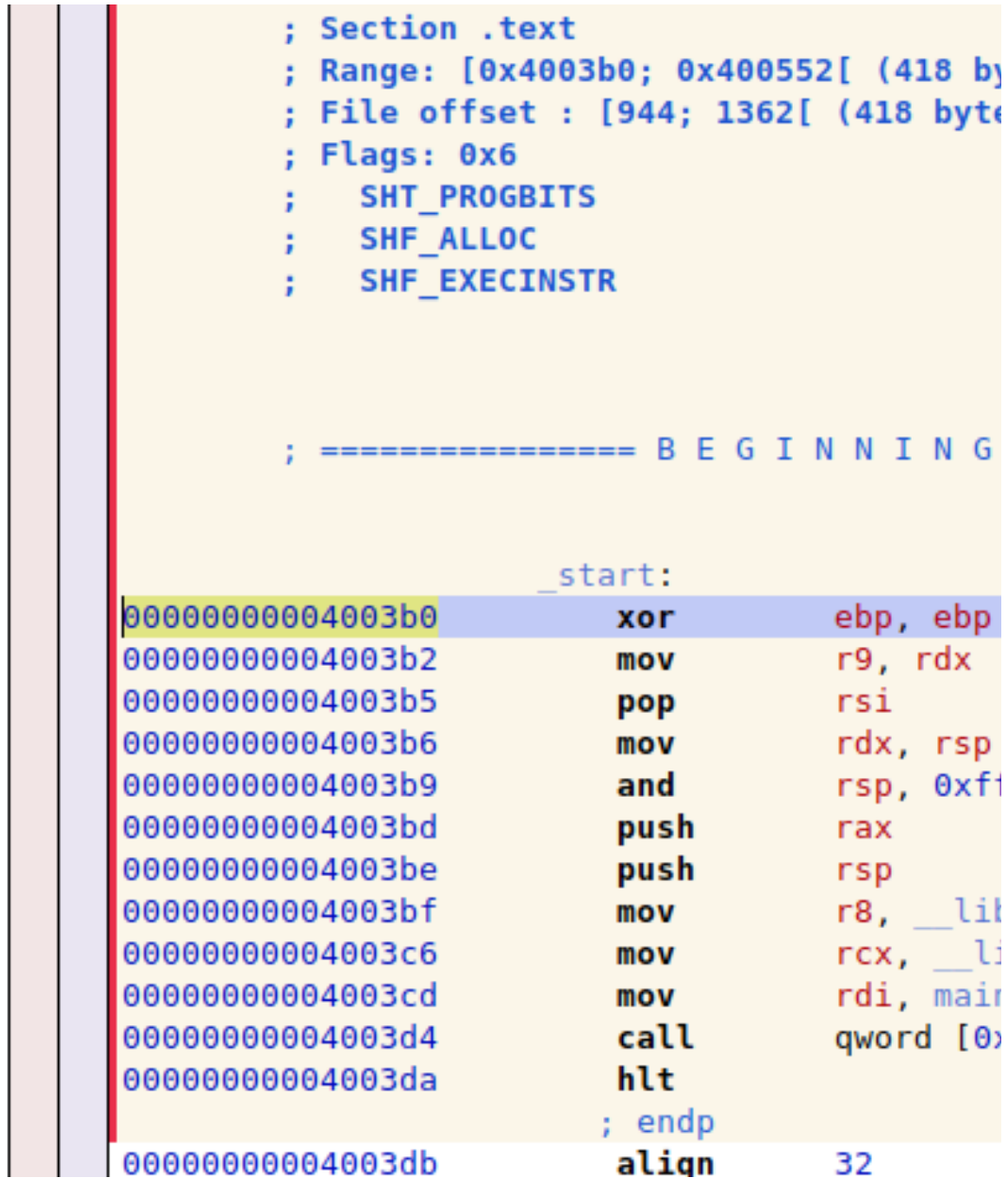
```
     0 SECTION LOCAL  DEFAULT   28
     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
     0 FUNC    LOCAL  DEFAULT   11 deregister_tm_
     0 FUNC    LOCAL  DEFAULT   11 register_tm_cl
     0 FUNC    LOCAL  DEFAULT   11 __do_global_dt
     1 OBJECT  LOCAL  DEFAULT   22 completed.7696
     0 OBJECT  LOCAL  DEFAULT   17 __do_global_dt
     0 FUNC    LOCAL  DEFAULT   11 frame_dummy
     0 OBJECT  LOCAL  DEFAULT   16 __frame_dummy_
     0 FILE    LOCAL  DEFAULT  ABS function2.c
     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
     0 OBJECT  LOCAL  DEFAULT   15 __FRAME_END__
     0 FILE    LOCAL  DEFAULT  ABS
     0 NOTYPE  LOCAL  DEFAULT   16 __init_array_e
     0 OBJECT  LOCAL  DEFAULT   18 _DYNAMIC
     0 NOTYPE  LOCAL  DEFAULT   16 __init_array_s
     0 NOTYPE  LOCAL  DEFAULT   14 __GNU_EH_FRAME
     0 OBJECT  LOCAL  DEFAULT   20 _GLOBAL_OFFSET
     2 FUNC    GLOBAL DEFAULT   11 __libc_csu_fin
     0 NOTYPE  WEAK   DEFAULT   21 data_start
    14 FUNC    GLOBAL DEFAULT   11 call2
     0 NOTYPE  GLOBAL DEFAULT   21 _edata
     0 FUNC    GLOBAL DEFAULT   12 _fini
     0 FUNC    GLOBAL DEFAULT  UND __libc_start_m
     0 NOTYPE  GLOBAL DEFAULT   21 __data_start
     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     0 OBJECT  GLOBAL HIDDEN    21 __dso_handle
     4 OBJECT  GLOBAL DEFAULT   13 _IO_stdin_used
   101 FUNC    GLOBAL DEFAULT   11 __libc_csu_ini
     0 NOTYPE  GLOBAL DEFAULT   22 _end
     2 FUNC    GLOBAL HIDDEN    11 _dl_relocate_s
    43 FUNC    GLOBAL DEFAULT   11 _start
     0 NOTYPE  GLOBAL DEFAULT   22 __bss_start
    34 FUNC    GLOBAL DEFAULT   11 main
     0 OBJECT  GLOBAL HIDDEN    21 __TMC_END__
    23 FUNC    GLOBAL DEFAULT   11 call
     0 FUNC    GLOBAL DEFAULT   10 _init
-playground $
```



Symbol Types

__dso_handle (OBJECT)





Symbol Types

_start (FUNC)

nm & strings

Symbol lister

- Fast and easy way to get symbols
- Symbol types
  - B, T, U, w, etc.
- Addresses

Strings

- Extract strings from a file

GDB is next.